Ollama API Manual

Complete Reference for n8n Integration

`Base URL: http://{private_ip}:11434`

1. Health Check

Confirm Ollama is running by sending a simple GET request.

Method	URL	Response
GET	http://{private_ip}:11434	Ollama is running

2. Complete API Endpoint Reference

Model Information

Method	Endpoint	Description
GET	/api/tags	List all locally available models
POST	/api/show	Show detailed info about a specific model
GET	/api/ps	List models currently loaded in memory

Text Generation

Method	Endpoint	Description
POST	/api/generate	Generate a completion from a prompt
POST	/api/chat	Chat with message history (conversation)

Embeddings

Method	Endpoint	Description
POST	/api/embed	Generate embeddings for text

Model Management

Method	Endpoint	Description
POST	/api/pull	Download a model from the Ollama library
POST	/api/push	Upload a model to the Ollama library
POST	/api/create	Create a model from a Modelfile
POST	/api/copy	Duplicate a model under a new name
DELETE	/api/delete	Delete a model

OpenAI-Compatible Endpoints

Ollama also exposes OpenAI-compatible endpoints, making it a drop-in replacement for many tools.

Method	Endpoint	Description
POST	/v1/chat/completions	OpenAI-compatible chat completions
POST	/v1/completions	OpenAI-compatible text completions
GET	/v1/models	OpenAI-compatible model listing
POST	/v1/embeddings	OpenAI-compatible embeddings

3. List Available Models

This is how you discover what models are installed on your Ollama server.

n8n HTTP Request Node Setup

Setting	Value
Method	GET
URL	http://{private_ip}:11434/api/tags
Body	None required

Example Response

```
{
  "models": [
    {
      "name": "llama3.2:latest",
      "model": "llama3.2:latest",
      "size": 2019393189,
      "details": {
        "family": "llama",
        "parameter_size": "3B",
        "quantization_level": "Q4_K_M"
      }
    }
  ]
}
```

Use the name field (e.g., llama3.2:latest) as the model identifier in all other endpoints.

4. Show Model Details

Setting	Value
Method	POST
URL	http://{private_ip}:11434/api/show

Request Body

```
{
  "model": "llama3.2"
}
```

Returns the Modelfile, template, parameters, and license for a model.

5. Generate Text (Single Prompt)

Setting	Value
Method	POST
URL	http://{private_ip}:11434/api/generate

Request Body

```
{
  "model": "llama3.2",
  "prompt": "Explain what a VPN is in one paragraph.",
  "stream": false
}
```

⚠ *Always set "stream": false when using n8n. Streaming returns multiple JSON chunks which n8n cannot parse. Setting it to false returns a single complete JSON response.*

Response

```
{
  "model": "llama3.2",
  "response": "A VPN, or Virtual Private Network, is a technology that...",
  "done": true,
  "total_duration": 5043500667,
  "eval_count": 87
}
```

Access the generated text in subsequent n8n nodes with: {{ $json.response }}

Optional Parameters

```
{
  "model": "llama3.2",
  "prompt": "Your prompt here",
  "stream": false,
  "system": "You are a helpful network engineer.",
  "format": "json",
  "options": {
    "temperature": 0.7,
    "top_p": 0.9,
    "num_predict": 500,
    "seed": 42
  },
  "keep_alive": "10m"
}
```

Parameter	Description
system	System message (overrides model default)
format	Set to "json" to force structured JSON output
options.temperature	Creativity (0 = deterministic, 1 = creative)
options.top_p	Nucleus sampling threshold

options.num_predict	Maximum tokens to generate
options.seed	Fixed seed for reproducible output
keep_alive	How long to keep model loaded ("5m", "1h", "0" to unload)

6. Chat (Conversation with History)

The most useful endpoint for building conversational workflows. Supports message history for context.

Setting	Value
Method	POST
URL	http://{private_ip}:11434/api/chat

Request Body

```
{
  "model": "llama3.2",
  "messages": [
    { "role": "system", "content": "You are a helpful IT support assistant." },
    { "role": "user", "content": "My laptop won't connect to WiFi." }
  ],
  "stream": false
}
```

Multi-Turn Example

```
{
  "model": "llama3.2",
  "messages": [
    { "role": "system", "content": "You are a helpful assistant." },
    { "role": "user", "content": "What is Docker?" },
    { "role": "assistant", "content": "Docker is a platform for..." },
    { "role": "user", "content": "How does it differ from a VM?" }
  ],
  "stream": false
}
```

Response

```
{
  "model": "llama3.2",
  "message": {
    "role": "assistant",
    "content": "Docker differs from VMs in that..."
  },
  "done": true
}
```

Access the response in n8n with: {{ $json.message.content }}

7. Generate Embeddings

Useful for semantic search, RAG pipelines, and similarity comparisons.

Setting	Value
Method	POST
URL	http://{private_ip}:11434/api/embed

```
{
```

```
  "model": "llama3.2",
  "input": "What is machine learning?"
}
```

You can also pass an array for batch embeddings:

```
{
  "model": "llama3.2",
  "input": ["First document", "Second document"]
}
```

8. Pull (Download) a New Model

Setting	Value
Method	POST
URL	http://{private_ip}:11434/api/pull

```
{
  "model": "mistral",
  "stream": false
}
```

Downloads the model from the Ollama library (ollama.com/library). May take several minutes.

Popular Models

Model	Size	Good For
llama3.2	3B	General purpose, fast
llama3.1:8b	8B	Strong general purpose
mistral	7B	General purpose, efficient
codellama	7B	Code generation
qwen2.5-coder	7B	Code generation
phi4	14B	Reasoning
deepseek-r1	Various	Reasoning with chain-of-thought
llava	7B	Vision (image + text)
nomic-embed-text	137M	Embeddings only

9. Copy and Delete Models

Copy a Model

```
POST http://{private_ip}:11434/api/copy

{
  "source": "llama3.2",
  "destination": "my-llama-backup"
}
```

Delete a Model

```
DELETE http://{private_ip}:11434/api/delete

{
  "model": "my-llama-backup"
}
```

10. Check Running Models

Shows which models are currently loaded in GPU/CPU memory.

Setting	Value
Method	GET
URL	http://{private_ip}:11434/api/ps

11. n8n Workflow Examples

Example 1: Simple Question-Answering

Workflow: Manual Trigger → HTTP Request

Setting	Value
Method	POST
URL	http://{private_ip}:11434/api/generate
Body Content Type	JSON

```
{
  "model": "llama3.2",
  "prompt": "Summarize the benefits of containerization in 3 bullet points.",
  "stream": false
}
```

Access the response in subsequent nodes with: {{ $json.response }}

Example 2: Dynamic Prompt from Webhook

Workflow: Webhook → HTTP Request → Respond to Webhook

Webhook receives a POST with: { "question": "What is Kubernetes?" }

```
{
  "model": "llama3.2",
  "messages": [
    { "role": "system", "content": "You are a DevOps expert. Be concise." },
    { "role": "user", "content": "={{ $json.body.question }}" }
  ],
  "stream": false
}
```

Respond to Webhook returns: {{ $json.message.content }}

Example 3: List Models Then Generate

Workflow: Manual Trigger → List Models (GET) → Generate (POST)

Node 1 lists models with GET /api/tags, then Node 2 uses the first model dynamically:

```
{
  "model": "={{ $json.models[0].name }}",
  "prompt": "Hello, what model are you?",
  "stream": false
}
```

Example 4: Email Summarizer

Workflow: Email Trigger (IMAP) → HTTP Request → Send Email (SMTP)

```
{
  "model": "llama3.2",
  "messages": [
    {
      "role": "system",
      "content": "Summarize the following email in 2-3 sentences. Include any action
items."
    },
    {
      "role": "user",
      "content": "={{ $json.text }}"
    }
  ],
  "stream": false
}
```

Example 5: Structured JSON Output

Force the model to respond with valid JSON for feeding into other n8n nodes.

```
{
  "model": "llama3.2",
  "prompt": "Extract the name, email, and company from this text:
    'Hi, I'm Sarah Chen from Acme Corp. Reach me at sarah@acme.com'
  ",
  "format": "json",
  "stream": false
}
```

Example 6: Using n8n Built-in Ollama Nodes

n8n has native Ollama integration nodes which handle the API calls for you:

1. Add an Ollama Chat Model node (under AI → Language Models)
2. Create credentials: Set the API URL to http://{private_ip}:11434
3. Connect it to a Basic LLM Chain or AI Agent node
4. Select your model from the dropdown (n8n auto-fetches the list from /api/tags)

Workflow: Chat Trigger → Basic LLM Chain (with Ollama Chat Model as sub-node)

12. Tips and Troubleshooting

⚠ stream: false is essential. The single most common issue with n8n + Ollama is forgetting this. Without it, Ollama streams multiple JSON objects and n8n gets a parse error.

Timeouts

Large models can take 30-60+ seconds to respond. In the n8n HTTP Request node, increase the timeout under Settings → Timeout (set to 120000ms or higher).

Model Naming

Models follow the format name:tag. If you omit the tag, it defaults to :latest. So llama3.2 is the same as llama3.2:latest.

Keep-Alive Behavior

By default, Ollama unloads a model from memory after 5 minutes of inactivity. Set "keep_alive": "1h" in your request to keep it loaded longer, or "keep_alive": "0" to unload immediately after the request.

Check Loaded Models

Hit GET /api/ps to see which models are currently in memory and how much VRAM they are using.

Docker Networking

If n8n is in Docker and Ollama is on the host, you may need to use host.docker.internal instead of localhost. If you are using a LAN IP like {private_ip} this is typically not an issue.

OpenAI Compatibility

Many tools that support the OpenAI API can point directly at Ollama using the /v1/ endpoints. Just set the base URL and use any string as the API key (Ollama does not require one).